(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 408 810 A1**

(12) # EUROPEAN PATENT APPLICATION

(21) Application number: 89307362.7

(51) Int. Cl.5: G06F 15/80, G06F 13/38

(22) Date of filing: 20.07.89

Amended claims in accordance with Rule 86 (2) EPC.

(43) Date of publication of application:
23.01.91 Bulletin 91/04

(84) Designated Contracting States:
DE GB NL

(71) Applicant: AKEBIA LIMITED
Lever House, 3 St. James's Road
Kingston Upon Thames, Surrey KT1 2BA(GB)

(72) Inventor: Thiel, Geoffrey Lawrence
27 Palace Road
East Molesey, Surrey KT8 9DJ(GB)
Inventor: Pontin, Paul Simon
30 Cedarcroft Road
Chessington, Surrey KT9 1RP(GB)

(74) Representative: Abnett, Richard Charles et al
REDDIE & GROSE 16 Theobalds Roadoad
London WC1X 8PL(GB)

(54) **Multi processor computer system.**

(57) A computer system (30) includes sixteen data processors (32a....32p) each connected to a communication bus (36). The communication bus (36) comprises a data bus (40) for carrying data, and an address bus (38) for carrying associated labelling information uniquely identifying the data. Each processor (32) includes read and write detectors (52) connected to the address bus (38) for detecting labelling information of data required by, or presently stored in, respectively, the data processor (32). A bulk memory (44) having similar read and write detectors (46) is connected to the communication bus (36). An address generator (32) supplies labelling addresses to the address bus (38). For each address, one processor (32) or the bulk memory (44) supplies the corresponding data to the data bus (40), and other processors and/or the bulk memory requiring the data read the data from the data bus (40). Data is transferred between processors and/or the bulk memory in this way. The address bus (38) and the read and write decoders (52, 46) are configured for multi-dimensional addressing.

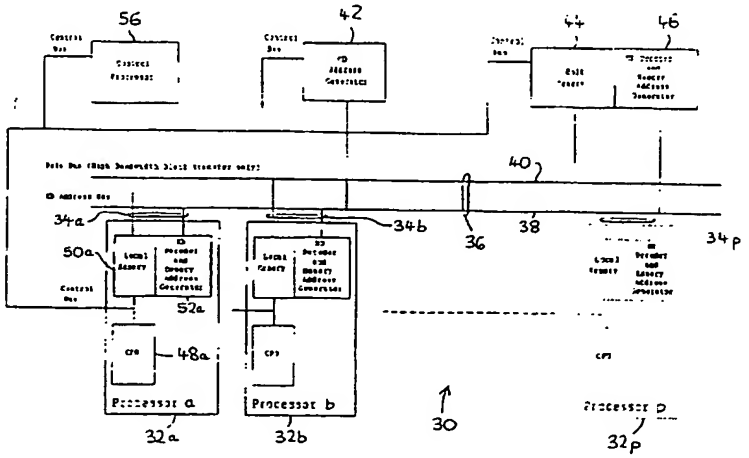


FIG 5

EP 0 408 810 A1

The present invention relates to the field of computer architecture for multi processor systems. In particular, the invention relates to systems in which the individual processors are interconnected by a common communication bus. Such a system is shown schematically for example in Figure 1 of the accompanying drawings. In Figure 1, each processor 20a, 20b, etc. is connected to the bus 22. Each processor has an area of local memory which the processor can access (local transfer) without using the bus 22. Data not held in the local memory for a respective processor is transferred by means of the bus (non local transfer).

We have appreciated that a problem with the above system is caused by the manner in which the system hardware supports software organised data. Computer memory hardware is normally linearly addressed as a one dimensional array of words. Computer software on the other hand requires a variety of data structures of which the most common is the data array. The problem arises from the way in which a multi-dimensional data array maps onto the memory hardware. Items which are logically closely related in the software maybe scattered throughout the linearly addressed memory hardware. Figure 2 illustrates the problem in the case of a 3 dimensional data array. A row of data in the X direction corresponds conveniently to a contiguous block in the memory hardware whereas a column of data in the Z direction is physically scattered throughout the entire memory. This effect can cause major delays within a parallel computer system because if a processor requires non-local data in any other direction than the X row direction the whole data array must be fetched to the local memory of the processor, which results in far more data being transferred than necessary. Alternatively, each element of the desired subset may be accessed individually creating many single word transfers which the hardware will not execute efficiently.

We have appreciated that another effect which leads to inefficiency in conventional systems takes place when several processors each require different but partially overlapping blocks of data. Figure 3 illustrates a simplified example where four processors each require approximately half of the available data. Some areas of the source data are required by all four processors whereas other areas are required by two processors or just one processor. In a conventional parallel processor system the whole of the data has to be transferred to each of the local memories of processors, which wastes memory space. Alternatively the desired data blocks have to be unpacked, i.e. the data is arranged as separate blocks for each processor, and sent individually to each of the processors in turn, which would waste processor time and communication bandwidth.

In the above examples it has been assumed that the data source is a single memory. The situation becomes more complex when the source data array is mapped across the local memories of a number of processors, and it has to be redistributed around the processors. This situation frequently arises from one algorithm stage to the next where the result data mapping from the first stage is not the same as the input data mapping required for the second. Figure 4 illustrates this situation where four processors hold data in their local memories in row format (Figure 4a) and the next algorithm stage requires the data in column format (Figure 4b). The resulting data movements are quite complex even in this example of just four processes (Figure 4c). In the above case a conventional system has to unpack and re-pack the data between the full array format and the sixteen subarrays illustrated. This is inefficient and wastes processor time.

The present invention is defined in the appended claims.

An embodiment of the invention will now be described by way of example with reference to the remaining figures of the accompanying drawings, in which:-

Figure 5 is a block diagram of an MIMD computer system;

Figure 6 is a block diagram showing in more detail an address generator shown in Figure 5;

Figure 7 shows an example of the format of an MD address;

Figure 8 is a block diagram showing in more detail an MD decoder and memory address generator shown in Figure 5; and

Figures 9, 10, 11 and 12 show formats of data transferred within the system in Figure 5.

Referring to Figure 5, a multiple instruction multiple data (MIMD) parallel computer system 30 includes sixteen data processors labelled 32a, 32b to 32p, respectively. For clarity, only the processors 32a, 32b and 32p are shown, however, the circuitry and connections for the processors 32a,b,p, are all similar, and it is to be assumed that the remaining processors are equivalent to those shown. In the present embodiment, the processors are 32-bit devices.

In an MIMD computer the individual processors operate independently, but often require data which is scattered throughout the system.

Each processor 32a,b,p is connected by means of respective link 34 to a communication bus 36. The communication bus 36 includes a multi-dimensional (MD) parallel address bus 38 which is 32-bits wide, and a data bus 40 which is 64 bits wide. An MD address generator 42 is connected to the address bus 38 to supply MD ad-

dresses to the bus which addresses act as labelling information uniquely identifying data on the data bus 40.

Each processor 32 comprises a respective central processing unit (CPU) 48, a respective local memory 50 and a respective associated MD decoder and memory address generator 52 to act as a read detector and a write detector. Each decoder generator 52 has programmable upper and lower read limits and upper and lower write limits for each of the dimensions of the MD address. In use, the decoder generator 52 decodes multidimensional addresses received from the MD address bus 38 and, if the decoded address falls with the programmed limits of the MD address, the address is translated into a linear address to access the memory 50.

Each local memory 50 is connected by data inputs/outputs to both the data bus 40 and to the respective CPU 48. The connection to the data bus 40 has a higher priority than the connection to the CPU, so that whenever a valid MD address is decoded by the respective MD decoder and memory address generator 52, data is transferred between the data bus 40 and the local memory 50. If the respective CPU is currently accessing the local memory 50, its access is temporarily paused while data transfer to or from the data bus 40 takes place.

The system also includes a bulk memory 44 built from dynamic RAM, and having its data input/output connected to the data bus 40. In the present embodiment, the bulk memory is 256 bits wide, ie. each MD address accesses 256 data bits. The memory 44 has an associated MD decoder and memory address generator 46 which is connected to the MD address bus 38 to act as a read detector and a write detector.

The decoder generator 46 is similar to the decoder generator 52 described hereinbefore, having programmable upper and lower limits for each of the dimensions of the MD address.

The computer system includes therefore a plurality of data processors 32 and a communication bus 36, each processor being connected to the communication bus 36. The communication bus 36 carries data and associated labelling information uniquely identifying the data. Each processor 32 includes a read detector 52 connected to the communication bus 32 for detecting labelling information of data required by the data processor and means 34, 52 for reading from the communication bus the data corresponding to the detected labelling information. Each processor also includes a write detector 52 connected to the communication bus 36 for detecting labelling information of data currently stored in the processor, and means 34, 52 for writing to the communication bus the data corresponding to the detected labelling information.

The processors 32, the MD address generator 42 and the bulk memory 44 are each connected to a control bus 54. A control unit processor 56 supplies control data to the control bus 54 to manage the computer system.

Referring to Figure 6, the MD address generator 42 comprises N+1 one dimensional address generators 60, where N is the maximum number of address dimensions that the hardware can accommodate, the lowest field being dimension zero. In the present embodiment, N=2, the addresses having dimensions 0, 1 and 2, and the corresponding address generators 60 being labelled 60a, 60b and 60c respectively. In other embodiments, more one dimensional address generators may be provided to allow for higher address dimensions, there being a respective address generator for each address dimension. The generator 42 also includes merge logic 62 to assemble the output from the address generators 60 as the MD address, and reload logic 64 to control increments between the generators 60. A loop counter 66 is connected to the reload logic to count through the number of addresses generated.

Each one dimensional address generator 60 includes a counter 70 having a COUNT input 72 and a RELOAD input 74, both fed from the reload logic 64, a reload value input 76 and an output 78. An initial value register 80 is connected to the reload value input 76. In use, when a reload signal appears at the RELOAD input 74, the counter 70 is reset to the value held in the initial value register 80. The counter output 78 is connected to the input of a comparator 82 which has a second input fed from a boundary value register 86, and an output 88 connected to feed to the reload logic. In use, when the value of the counter output 78 is equal to the value held in the boundary value register 86, the comparator 82 outputs an EQUAL signal to the reload logic. The counter output 78 is also connected to the input of a shift and mask network 90, with an associated shift value register 92 and an associated value register 94. In use, the network 90 shifts and masks the value of the counter output 78, depending on the values held in the registers 92 and 94, to form one field of the MD address. The output of the shift and mask network 90 is connected to the merge logic 62.

In use, the MD address generator 42 generates a sequence of MD addresses as follows:

The Control Processor 36 initialises the initial value, boundary value, shift value and mask value registers 80,86,92,94, respectively and counter 70 in each of the one dimensional address generators 60 and also the loop counter 66. After each cycle the COUNTO signal increments the dimension 0 counter 70a and decrements the loop counter 66. If

any of the count values becomes equal to its corresponding boundary value then the EQUAL signal for that dimension is asserted. This in turn causes a RELOAD signal back to its own counter and a COUNT signal to the next highest dimension. The counter output of each dimension is shifted and masked to form the individual address fields, these are then concatenated by the merge logic 62 to form the MD Address output. The loop countrer 66 records the total number of addresses generated and stops the process at the end.

Figure 7 shows the format at a typical MD address produced by MD address generator 42. It comprises the concatenation of a number of one dimensional address fields. The overall number and width of each address field is controlled by the control processor 56, which controls the shift and mask networks 90 of each one dimensional address generator 60.

Referring to Figure 8, each MD decoder and memory address generator 46,52a,b to 52p includes N+1 linear decoders 100a,b,c. connected in parallel to an MD address bus input 102. The MD address bus input 102 of each MD decoder is connected to the address bus (38 in Figure 5). Each decoder 100 decodes a respective field of the MD address.

Each decoder 100 includes a shift and mask network 104 having an input connected to the MD address input 102, an associated shift value register 106 and an associated mask value register 108. In use, the network 104 extracts only certain bits of the MD address depending on the value held in the mask value register 108, and shifts the extracted bits down in significance depending on the value held in the shift value register 106.

The output from the network 104 is connected to the input of a second comparator 110, which has another input connected via a switch (not shown) to one of a read lower boundary register 112 and a write lower boundary register 114. The output from the second comparator 110 is a logical 1 (high) whenever the output from the network 104 is equal to or greater than the value in the selected lower boundary register. The output from the network 104 is also connected to the input of a third comparator 116 which has another input connected via a switch (not shown) to one of a read upper boundary register 118 and a write upper boundary register 120. The output from the third comparator 116 is a logical 1 (high) whenever the output from the network 104 is equal to or less than the value in the selected upper boundary register. The output from the second and third comparators 110,116 respectively are connected to respective inputs of an AND gate 122. The output 124 from the AND gate 122 is the respective field decode output for the linear decoder, e.g. the output 124a is the field decode output for the linear decoder 100a which decodes the dimension zero field of the MD address. Each r spective output 124 is a logical 1 (high) only when the value of the respective field of the MD address lies between the lower and upper values stored in the selected lower and upper boundary value registers for the respective field decoder 100.

The outputs 124 from the decoders 100 are connected to respective inputs of an N+1 input AND gate 126. The output 128 from the AND gate 126 is the MD decode output from the MD decoder, and is a logical 1 (high) only when the value of each field in the MD address lies between the lower and upper boundary values for the respective address field.

Figure 9 shows an example of the decoding of a two dimensional address (y,x). The upper and lower boundary registers for the dimension zero (x) field of the address contain the values Xmax and Xmin, respectively. The upper and lower boundary registers for the dimension one (y) field of the address contain the values Ymax and Ymin, respectively. A decode signal will appear at the output 128 only when the value of x lies between Xmin and Xmax, and the value of y lies between Ymin and Ymax.

Referring again to Figure 8, each of the MD decoder and memory address generators 46,52 also includes a linear address generator 130 for generating a local memory location address from a decoded MD address. The linear address generator 130 includes N+1 field incrementers 132 each comprising a respective field increment register 134 connected to one input of a respective parallel multiplier 136. The output from the respective shift and mask network 104 is connected to another input of the parallel multiplier 136. The output from the multiplier 136 is taken as the output from the respective field incrementer 132, and each output is connected to a respective input of a parallel adder 138. The adder 138 also has an input connected by a switch (not shown) to one of a read array base address register 140 and a write array base generator 142. In use, each incoming decoded field address from the respective shift and mask networks 104 is multiplied in a respective field incrementer 132 by the increment value stored in the respective field increment register 134. The resulting product values are then combined in the adder together with the base address value stored in the read or write base address register. The output from the adder 138 corresponds to a linear address of a local memory location.

If Da is the value stored in the dimension zero field increment register 134a, and Db is the value stored in the dimension one field register 134b, and the array base addrerss is BP, the linear

memory address output from the MD decoder and memory address generator will be of the form

Memory address = BP + (Aa x Da) + (Ab x Db) + ...

where Aa, Ab, etc. are the values of the decoded field addresses for the dimension zero, dimension one, etc., respectively fields of the MD address.

Figure 10a shows a diagrammatic representation of a two dimensional array to be stored in memory. Each element in the array is addressable as an (y,x) co-ordinate. The array extends between the value O and Xlen in the x direction, and between the values 0 and Ylen in the y direction. To store this array in memory, the values of the variables in the above equation would be:

Da = Dx = 1

Db = Dy = Xlen + 1

and Address P(x,y) = BP + x + y (Xlen + 1). The way in which the elements of the array are stored in the memory is shown in Figure 10b, the base address BP corresponding to the position of the element P(O,O).

Referring to Figure 5, the data bus 40 is made as wide as is convenient, wider than the length of a data word. A highly parallel bus achitecture, known in itself, is used to achieve high bus performance. Data transfers on the bus are resticted to a minimum block size corresponding to the width of the data bus. The bus cycle time is made as short as possible within the constraints of available technology and system cost.

In the present embodiment, the data bus 40 is 64 bits wide and the bulk memory 44 is 256 bits wide. The period of each MD addrss cycle is 320ns, and during each MD access the 256 bits are transferred to the data bus as four 64-bit wide 'bus words'. Each bus word is transferred on a 'data cycle' of period 80ns (320ns/4). The effect of such a configuration is to increase the memory bandwidth, each address access comprising a block of contiguous data many bytes long. At the same time, the period of each MD address cycle is sufficient to allow the normally slower MD address decoding logic to function.

Each processor 32 has a further shift and mask network (not shown) at its input/output connection to the data bus 40. The shift and mask network can selectively access data on the data bus 40 to a resolution of 8 bits, in effect adding a further addressing stage of the data. The shift and mask network is controlled by the MD decoder and linear address generator to enable data to be read or written within the contiguous block.

Owing to the time required by the MD address decoders 46, 52 to decode. and compare the MD address with the read and write, upper and lower boundary values, the data cycles during which data is transferred on the data bus 40 are delayed in time behind the corresponding address cycle of the MD address on the MD address bus 38. In this way, the associated address and data information are pipelined.

Figure 12 shows an example of the transfer of address and data information on the communication bus. Referring to figure 12, address and data information is transferred on the communication bus in syncronisation with continuous bus cycles 160a, 160b, 160c, etc. (each of period 320ns). An address cycle 162a coincides with a bus cycle 160a. The corresponding data cycles 164a appear on the data bus during the subsequent bus cycle 160b. During this subsequent bus cycle 160b, a next MD address cycle 162b occurs, but the data cycles 164b carrying the data information associated with the MD address cycle 162b do not appear on the data bus until the further bus cycle 160c. Thus, the associated address and data cycles are pipelined. As explained hereinbefore, such a configuration allows efficient operation of both the address bus 38 and the data bus 40 while still allowing a pause (320ns) in which the MD address decoding logic and memory access can operate.

In use, the control processor 56 provides initialisation and run-time control for all parts of the computer system 30 by means of the control bus 54. Non-local data transfers, i.e. inter processor transfers and processor/bulk memory transfers, take place by means of the data bus 40 and the MD address bus 38. All non-local data transfers take the form of Direct Memory Access (DMA) block transfers.

During non-local data transfers, the MD address generator 42 supplies a sequence of MD address to the MD address bus 38 under the control of the control processor 56. Each address is supplied to the address bus during an address cycle. Within the MD decoders, each address cycle is split into a write cycle followed by a read cycle.

During the write cycle, the MD address is compared with the boundary values stored in the upper and lower write boundary registers 120, 114 respectively. If the MD address falls within all the boundary values for the particular decoder, the MD decode signal for that decoder is asserted, and a linear memory write address is obtained.

During the subsequent read cycle, each MD decoder and memory address counter compares the MD address fields with the boundary values held in each of the upper and lower read boundary registeres 118, 112 respectively. If the MD address falls within all of the boundary values for the particular decoder, the MD decode signal for that decoder is asserted, and a linear read address is obtained.

If the MD address does not fall within the read or write, upper and lower boundary address for the

particular decoder, no MD decode signal is asserted, and the output from the MD decoder and linear address generator remains dormant.

Depending on the respective outputs of the MD decoders 52 in the processors 32, and the MD decoder 46 of the bulk memory 44, each processor and the memory will, independently of one another, write, read or ignore data on the data bus. For there to be data on the databus 40, one of the processors 52 or bulk memory 44 must write data to the data bus. Only one such device should write data to the data bus for any given MD address. The hardware must be protected against bus contention in the event that two or more devices attempt to write data simultaneously. For any given MD address, one or more of the processors 32 and/or the bulk memory 44 may read data from the data bus 40 simultaneously.

Therefore, during each bus cycle, data may be both read from and written to different areas of the same memory. In this event, the respective MD decoder and memory address generator may operate to address independent read and write arrays. This is achieved by setting the values of the respective read array base address register 140 and the respective write array base address register 142 to point to different base addresses for the read and write arrays.

In the embodiment described above, data stored in an array may be transferred and distributed across a number of processor memories. By programming the read boundary values of the desired array blocks for each processor, the transfer of the data can be accomplished by the system in a single pass through the array addresses. During the transfer, each processor recognises data it should read by the MD address being between its preprogrammed boundary addresses, and the recognised data is read into the processor automatically. Where several processors require overlapping blocks of data, the respective boundary registers are set up with overlapping address fields. Each processor can read the data independently of the other processors, and the appropriate data is distributed to each processor during the single pass.

The source data for the array may also be distributed across a number of processors. By programming the write boundary values of the source data held by each processor, the appropriate source data will be transferred automatically from the respective processor to the data bus when the MD address is between the write boundary values for the respective processor.

Since, as explained above, both source and destination arrays may be distributed across a number of processors, the system allows the reordering or re-distribution of such data arrays between the processors, using only a single pass through the array addresses. Each processor is set up with the write boundary addresses for the data it currently holds, and the read boundary addresses for the data it is to read. During the transfer, the appropriate data for each MD address is written from a processor to the data bus, and the data is then read into the appropriate destination processor or processors. The redistribution of data between the processors and the bulk memory may also be accomplished using a similar transfer.

Referring to Figure 11, the redistribution of data can be illustrated by an example in which a two stage algorithm requires a redistribution of data between the first and second stages. A typical algorithm is a two dimensional Fast Fourier Transform (FFT). The data consists of 512 by 512 complex numbers which is stored in a three dimensional data array of size 2 x 512 x 512 (512K words). The 2D FFT algorithm comprises 512 one dimensional FFTs executed on the rows of data followed by 512 one dimensional FFTs executed on the columns of data. During the first phase of the algorithm accesses only take place along the direction of the rows and similarly in the second phase of the algorithm data accesses take place along the direction of the columns. The algorithm can therefore be split between the 16 processors very easily by giving each one 32 rows of data to transform in the first phase (Figure 11a) and 32 columns of data to transform in the second phase (Figure 11b). Between the first and second algorithm phases the Communication Bus is used to move the data between two buffers achieving a redistribution from row format to column format at the same time. Figure 11c shows the 256 data fragments which are automatically routed between the 16 processors by the Communication Bus. Sixteen of these actually have the same source and destination processor although the data still gets copied from one buffer to the other.

Although in the embodiment described above the computer system includes sixteen processors 32, it will be appreciated that in other embodiments a similar architecture may use more or less processors, for example two processors, or thirty-two processors. In the preferred embvodiment, the architecture allows new processors to be "plugged in" to the communication bus, or faulty or dormant processors to be switched out. When adding or removing processors, the software for the control unit processor has only to be reloaded and supplied with updated information of which processors are in use.

It will be appreciated that in the embodiment described above, the hardware directly supports information organised in multi-dimensional data arrays. Individual processors can conveniently ac-

cess any part of the data arrays by means of the communication bus, even though the data in the array may itself be distributed across the many physically separate memories of the system. The system may execute efficiently a variety of different algorithms, including those requiring a high proportion of data interchange between the processors. The method of access, the transfer latency and transfer speed are independent of the physical location of the data. The programming of the system in order to use the parallel hardware resources efficiently therefore requires litte additional effort compared to programming a single processor system.

It will also be appreciated that in the embodiment described above, each processor receives precisely the subset of data it requires, thereby optimising the usage of local memory. When several processors require some part, or all, of the same data, they each receive the appropriate parts during the same access cycle, thereby making efficient use of the available communication bandwidth.

It will be appreciated further that in the embodiment described above, data organised in multi-dimensional arrays is moved within the system while it is still in the multi-dimensional array format. The data does not have to be "unpacked" into a linear format, and then "repacked" into an array format in each processor. Data can also be transferred from the system to an external device while still in the multi-dimensional format.

A computer system in accordance with the invention is for example suitable in applications such as image processing, computer graphics, computer aided design and scientific computing.

Claims

1. A computer system including a plurality of data processors and a communication bus, each data processor being connected to the communication bus, the communication bus carrying data and associated labelling information uniquely identifying the data, each processor including a read detector connected to the communication bus for detecting labelling information of data required by the data processor and means for reading from the communication bus the data corresponding to the detected labelling information.

2. A computer system according to claim 1, wherein each data processor includes a write detector connected to the communication bus for detecting labelling information of data currently stored in the data processor, and means for writing to the communication bus the data corresponding to the detected labelling information.

3. A computer system according to claim 1 or 2, wherein the communication bus comprises a data bus for carrying the data and an address bus for carrying the associated labelling information.

4. A computer system according to claim 1,2 or 3, further comprising an address generator connected to the address bus for generating the labelling information.

5. A computer system according to claim 4, wherein the address generator generates multi-dimensional address information.

6. A computer system according to claim 1,23,4 or 5, further comprising a bulk memory connected to the communication bus, the bulk memory including a read detector connected to the communication bus for detecting labelling information of data to be stored in the memory and means for reading from the communication bus the data corresponding to the detected labelling information, the bulk memory also including a write detector connected to the communication bus for detecting labelling information of data currently stored in the bulk memory and means for writing to the communication bus the data corresponding to the detected labelling information.

7. A computer system according to any of claims 4 to 6, wherein the read detector includes a lower boundary register for each address dimension for holding a lower read boundary value therefor, an upper boundary register for each address dimension for holding an upper read boundary value therefor, and wherein the detector generates an output decode signal whenever the value of each address dimension is between the respective upper and lower read boundary values for that address dimension.

8. A computer system according to any of claims 4 to 7, wherein the write detector includes a lower boundary register for each address dimension for holding a lower write boundary value therefor, an upper boundary register for each address dimension for holding an upper write boundary value therefor, and wherein the detector generates an output decode signal whenever the value of each address dimension is between the respective upper and lower write boundary values for that address dimension.

9. A computer system according to any of claims 4 to 8, wherein the address generator generates a sequence of adjacent addresses, the sequence corresponding to a block of data.

10. A computer system according to claim 9, further comprising a control unit processor connected to each of the data processors and the address generator by means of a control bus, the control unit processor controlling the read and write recognition circuits for each processor and the address sequences for the address generator.

11. A computer system according to any of claims 4 to 10, wherein each data processor includes a central processing unit, a local memory and an address decoder and linear memory address generator, the memory address generator generating a local memory address from the labelling information on the address bus.

12. A computer system according to claim 11, wherein the address decoder and linear memory address generator includes the read detector and the write detector, and for each address generated on the communication bus, the decoder and linear memory address generator operates on a write cycle during which the write detector functions, followed by a read cycle during which the read detector functions.

13. A method of data communication in a computing system including a plurality of data processors and a communication bus, each data processor being connected to the communication bus, wherein data and associated labelling information uniquely identifying the data are supplied to the communication bus, and each processor detects labelling information of data it requires, and reads from the communication bus the data corresponding to the detected labelling information.

14. A method according to claim 13, wherein labelling information is supplied to thecommunication bus for identifying data to be supplied to the communication bus, and a processor holding the data corresponding to the labelling information detects the labelling information as labelling information of the data, and supplies the data to the communication bus.

Amended claims in accordance with Rule 86(2) EPC.

1. A computer system (30) including a plurality of data processors (32) and a communication bus (36), each data processor being connected to the communication bus, the communication bus carrying data and associated labelling information uniquely identifying the data, each processor including a programmable read detector (52) connected to the communication bus for detecting labelling information of data required by the data processor and means (34, 52) coupled to the read detector for reading from the communication bus the data corresponding to the detected labelling information, characterised in that each processor further includes a programmable write detector (52) connected to the communication bus for detecting labelling information of data currently stored in the data processor, the write detector being programmable independently of the read detector, and means (34, 52) coupled to the write detector for writing to the communication bus the data corresponding to the detected labelling information.

2. A computer system according to claim 1, wherein the communication bus comprises a data bus (40) for carrying the data and an address bus (38) for carrying the associated labelling information.

3. A computer system according to claim 2, wherein the data bus (40) comprises a parallel bus, the width of which is greater than the size of a data word, whereby data can be communicated as a plurality of data words in parallel.

4. A computer system according to claim 2 or 3, further comprising an address generator (42) connected to the address bus (40) for generating the labelling information.

5. A computer system according to claim 4, wherein the address generator (42) generates multi-dimensional address information.

6. A coputer system according to claim 1, 2, 3, 4 or 5, further comprising a bulk memory (44) connected to the communication bus, the bulk memory including a read detector (46) connected to the communication bus for detecting labelling information of data to be stored in the memory and means for reading from the communication bus the data corresponding to the detected labelling information, the bulk memory also including a write detector (46) connected to the communication bus for detecting labelling information of data currently stored in the bulk memory and means for writing to the communication bus the data corresponding to the detected labelling information.

7. A computer system according to any of claims 4 to 6, wherein the read detector includes a lower boundary register (112) for each address dimension for holding a lower read boundary value therefor, an upper boundary register (118) for each address dimension for holding an upper read boundary value therefor, and wherein the detector generates an output decode signal (128) whenever the value of each address dimension is between the respective upper and lower read boundary values for that address dimension.

8. A computer system according to any of claims 4 to 7, wherein the write detector includes a lower boundary register (114) for each address dimension for holding a lowre write boundary value therefor, an upper boundary register (120) for each address dimension for holding an upper write boundary value therefor, and wherein the detector generates an output decode signal (128) whenever the value of each address dimension is between the respective upper and lower write boundary values for that address dimension.

9. A computer system according to any of claims 4 to 8, wherein the address generator generates a sequence of adjacent addresses, the sequence corresponding to a block of data.

10. A computer system according to claim 9, further comprising a control unit processor (56) connected to each of the data processors and the address generator by means of a control bus (54), the control unit processor controlling the read and write detectors for each processor and the address sequences for the address generator.

11. A computer system according to any of claims 4 to 10, wherein each data processor includes a central processing unit (48), a local memory (50) and an address decoder and linear memory address generator (52), the memory address generator generating a local memory address from the labelling information on the address bus.

12. A computer system according to claim 11, wherein the address decoder and linear memory address generator (52) includes the read detector and the write detector, and for each address generated on the communication bus, the decoder and linear memory address generator operates on a write cycle during which the write detector functions, followed by a read cycle during which the read detector functions.

13. A method of data communication in a computing system including a plurality of data processors and a communication bus, wherein data and associated labelling information are supplied to the communication bus and each processor is programmed with read detection information of labelling information of data it requires, and write detection information of labelling information of data it currently stores, and wherein each processor detects labelling information of data corresponding to the read detection information, and reads from the communication bus the associated data, characterised in that the read and write detection information is programmed independently, and in that each processor detects labelling information of data corresponding to the write detection information, and supplies the associated data to the bus.

14. A method of data communication according to claim 13, characterised in that the associated labelling information comprises multi-dimensional address information.

15. A method of data communication according to claim 14, characterised in that the read detection information corresponds to a contiguous block of data, and wherein the write detection information corresponds to a contiguous block of data.

16. A method of data communication according to claim 14 or 15, characterised in that for each address of labelling information, the processors operate on a write cycle during which data is supplied to the bus, followed by a read cycle during which data is read from the bus.

17. A data processor (32) for use in a computer system in accordance with claim 1, the system including a communication bus (36) carrying data and associated labelling information uniquely identifying the data, the data processor comprising means (34) for coupling the processor to the comunication bus, a programmable read detector (52) for detecting labelling information on the communication bus of data required by the data processor, means (34, 52) coupled to the read detector for reading from the communication bus the data corresponding to the detected labelling information, and means (48, 50) for processing and storing data read from the communication bus, characterised in that the processor includes a programmable write detector (52) for detecting labelling information on the communication bus of data stored by the data processor, the write detector being programmable independently of the read detector, and menas (34, 52) coupled to the write detector for writing to the communication bus the data corresponding to the detected labelling information.

18. A multi processor system, comprising a communication bus for carrying data and associated labelling information uniquely identifying the data and a plurality of data processors coupled to the communication bus, each data processor including read detector means coupled to the communication bus for detecting labelling information of data required by the data processor, means coupled to the read detector means for reading from the communication bus the data corresponding to the detected labelling information, means coupled to the communication bus for detecting labelling information of data currently stored in the data processor, and means coupled to the write detector means for writing to the communication bus the data corresponding to the detected labelling information, characterised by means for controlling the read and write detector means, to operate on the labelling information firstly on a write cycle during which data can be written to the communication bus, followed by a read cycle, during which data can be read from the communication bus.

19. A method of data communication in a computing system, the system comprising a communication bus and a plurality of data processors coupled to the communication bus, the method comprising supplying labelling information to the communication bus, the labelling information being associated with and uniquely identifying data to be carried by the communication bus, programming each data processor with read detection information of labelling information corresponding to data it requires and programming each data processor with write detection information of labelling information corresponding to data it currently stores, characterised in that each processor operates on the labelling information on the communication bus, firstly on a write cycle during which each data processor de-

tects labelling information of data corresponding to the write detection information, and supplies to the communication bus data corresponding to detected labelling information, and each processor operating subsequently on a read cycle during which each data processor detects labelling information of data corresponding to the read detection information, and reads from the communication bus data corresponding to detected labelling information.

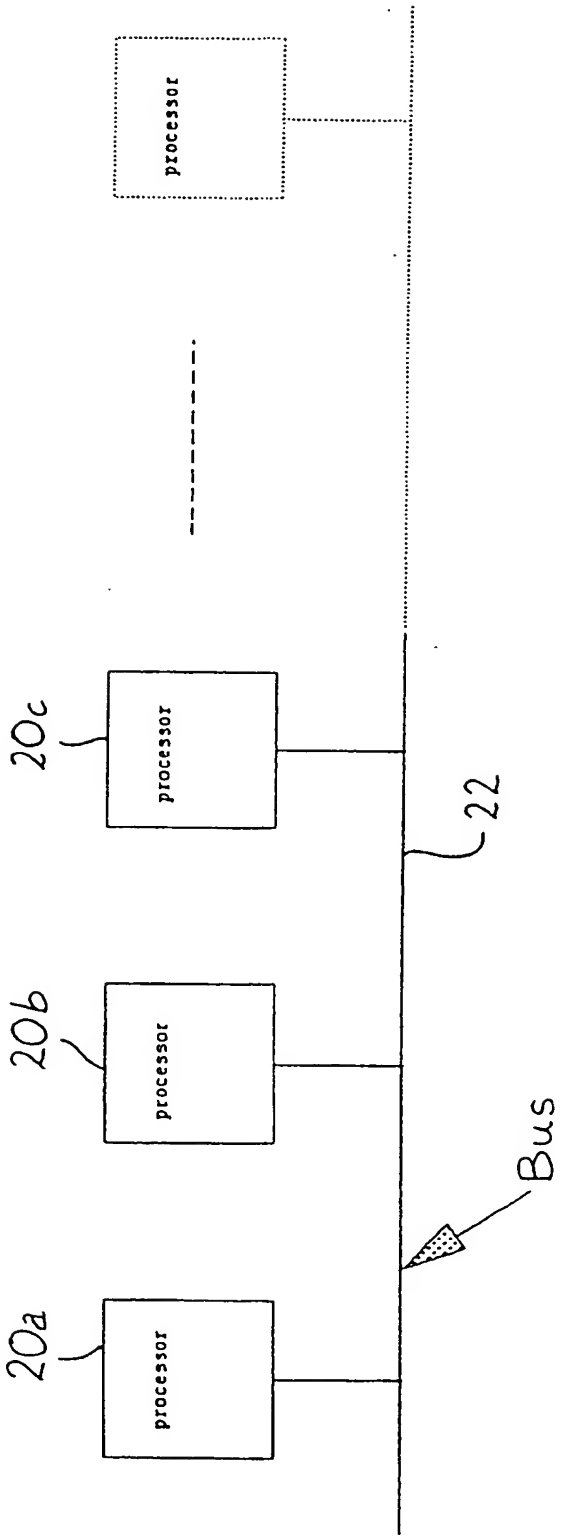

processor
processor
processor
processor
20c
20b
20a
22
Bus

FIG. 1

3 Dimensional Axis

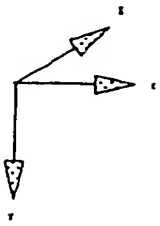


Figure
3 Dimensional Array
DIMENSION (3,3,3)

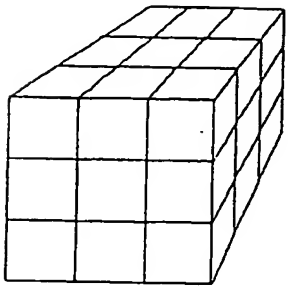


Figure
Physical Memory
27 Linearly Addressed Words

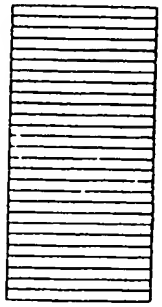


Figure
A row of data in the X direction

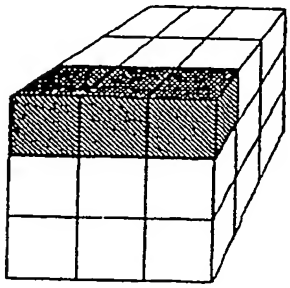


Figure
Rows in X are located in 3 sequential memory locations

Figure
A column of data in the Z direction

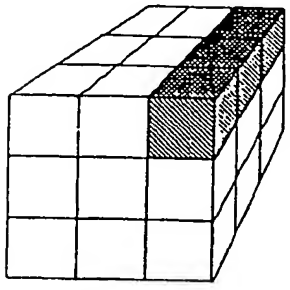


Figure
Columns in Z are scattered 9 words apart

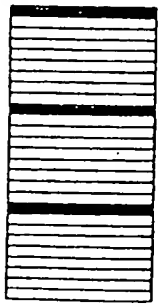


FIG. 2

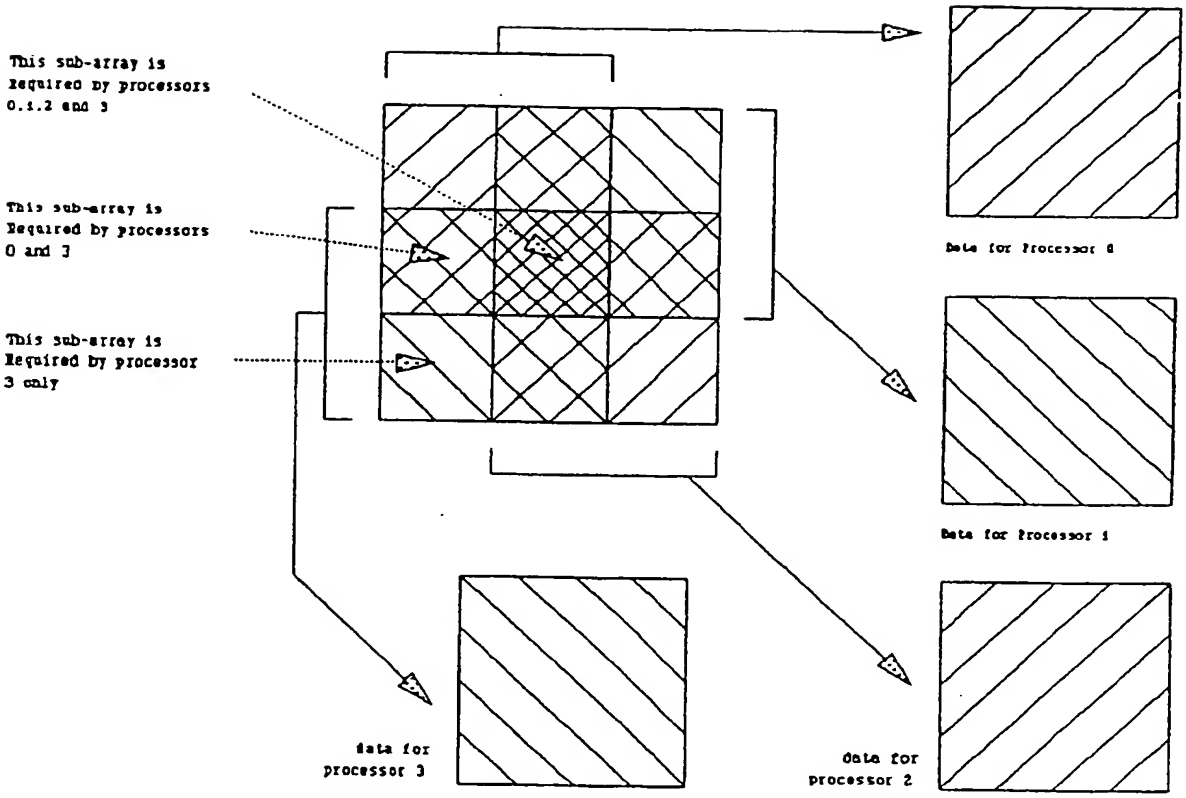

This sub-array is
Required by processors
0,1,2 and 3
This sub-array is
Required by processors
0 and 3
This sub-array is
Required by processor
3 only
Data for Processor 0
Data for Processor 1
data for
processor 3
data for
processor 2

FIG. 3

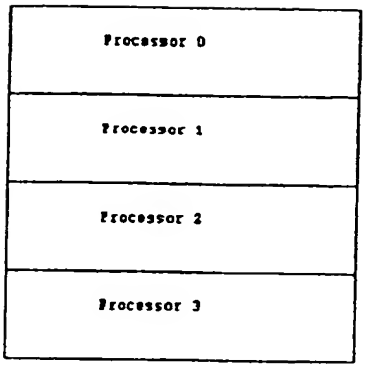


Fig 6a SOURCE FORMAT Data is in row format and distributed between the local memories of the four processors.

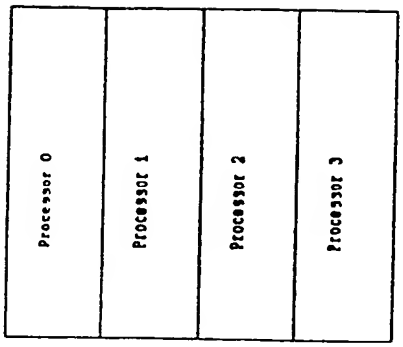
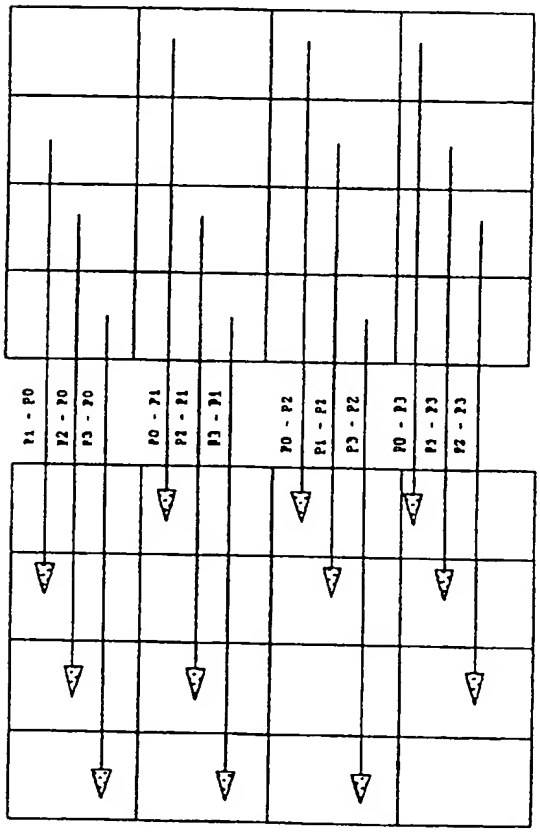


Fig 6b DESTINATION FORMAT Data is re-distributed in column format



Source Mapping

Destination Mapping

Fig 6c Data movements required to sort data from row format to column format.
Note that as the data is two dimensional normally the processors would have to unpack and re-pack the sixteen data sub-arrays.

Fig.4

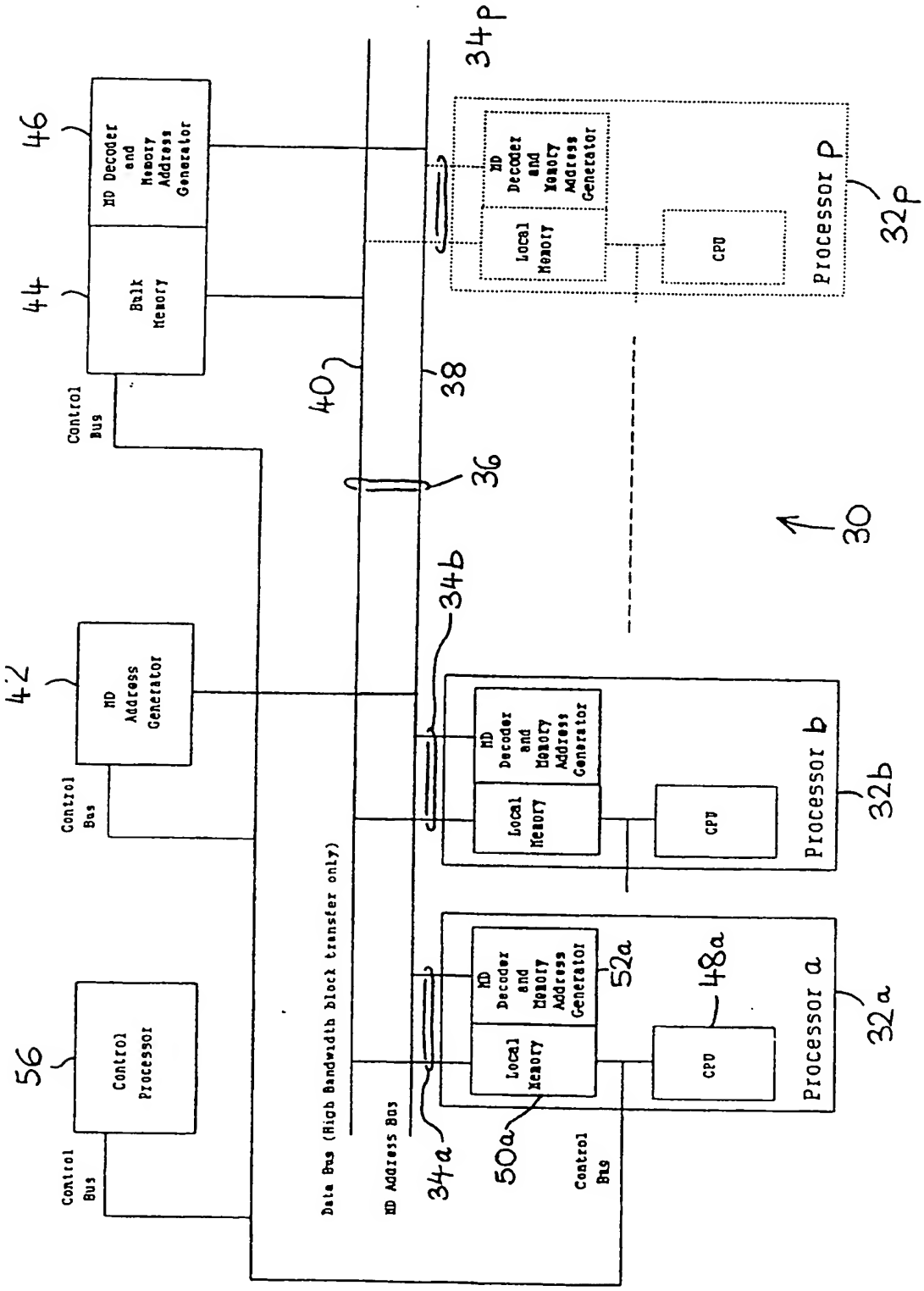
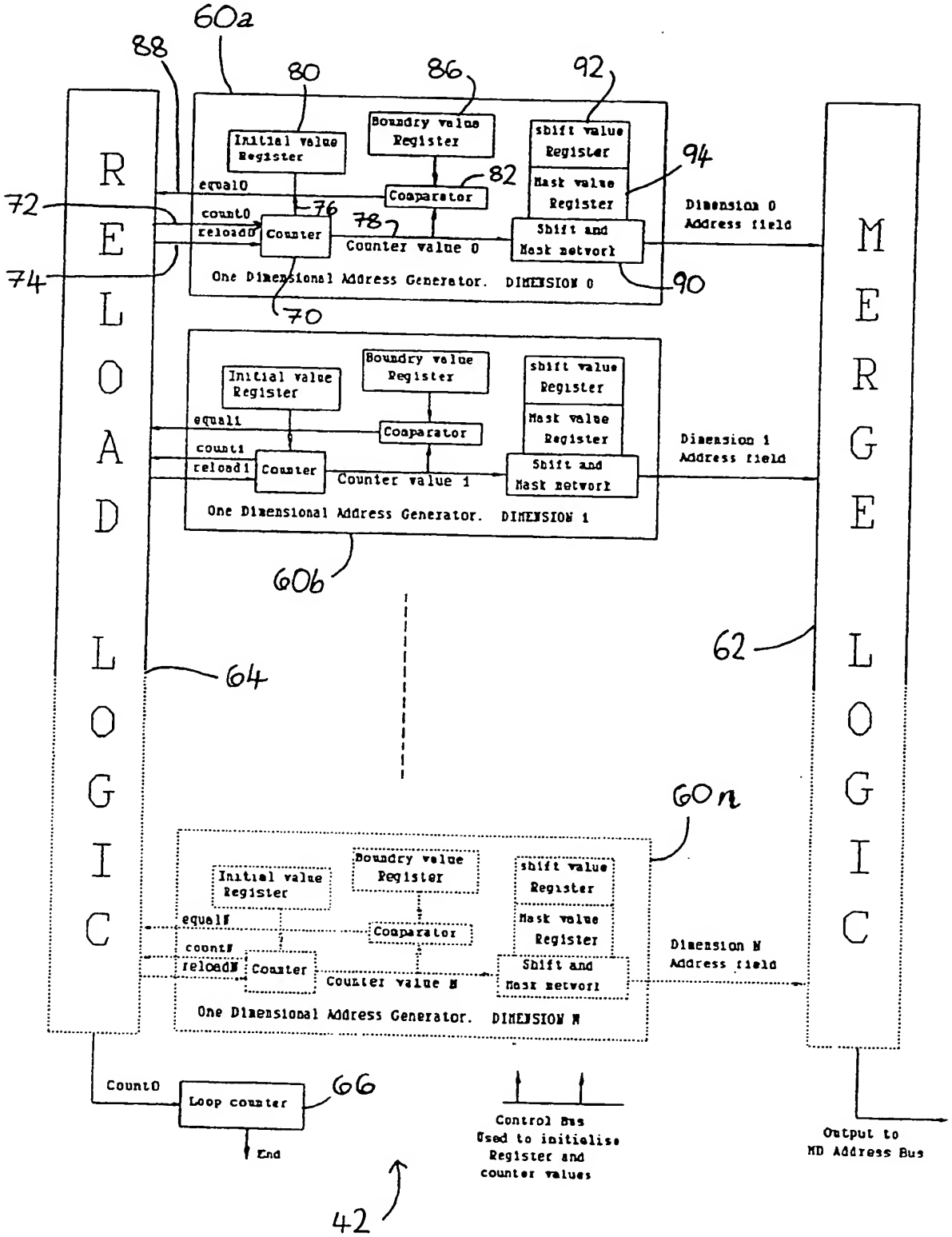
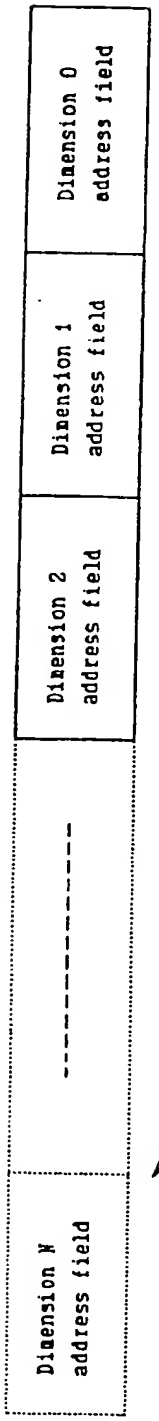

46
MD Decoder and Memory Address Generator
44
Blk Memory
Control Bus
34p
MD Decoder and Memory Address Generator
Local Memory
CPU
Processor P
32p
40
38
36
30
42
MD Address Generator
Control Bus
34b
MD Decoder and Memory Address Generator
Local Memory
CPU
Processor b
32b
56
Control Processor
Control Bus
Data Bus (High Bandwidth block transfer only)
MD Address Bus
34a
50a
MD Decoder and Memory Address Generator
52a
Local Memory
48a
CPU
Processor a
32a
Control Bus

FIG. 5


60a
88
80
86
92
Initial value Register
Boundry value Register
shift value Register
Mask value Register
94
82
equal0
count0
reload0
Counter
76
78
Comparator
Counter value 0
Shift and Mask network
Dimension 0 Address field
72
74
90
70
One Dimensional Address Generator. DIMENSION 0
R E L O A D L O G I C
M E R G E L O G I C
Initial value Register
Boundry value Register
shift value Register
Mask value Register
equal1
count1
reload1
Counter
Comparator
Counter value 1
Shift and Mask network
Dimension 1 Address field
One Dimensional Address Generator. DIMENSION 1
60b
64
62
60n
Initial value Register
Boundry value Register
shift value Register
Mask value Register
equalN
countN
reloadN
Counter
Comparator
Counter value N
Shift and Mask network
Dimension N Address field
One Dimensional Address Generator. DIMENSION N
Count0
Loop counter
66
End
42
Control Bus Used to initialise Register and counter values
Output to MD Address Bus



Note: Both the number of dimensions and the width of each field is variable

FIG. 7

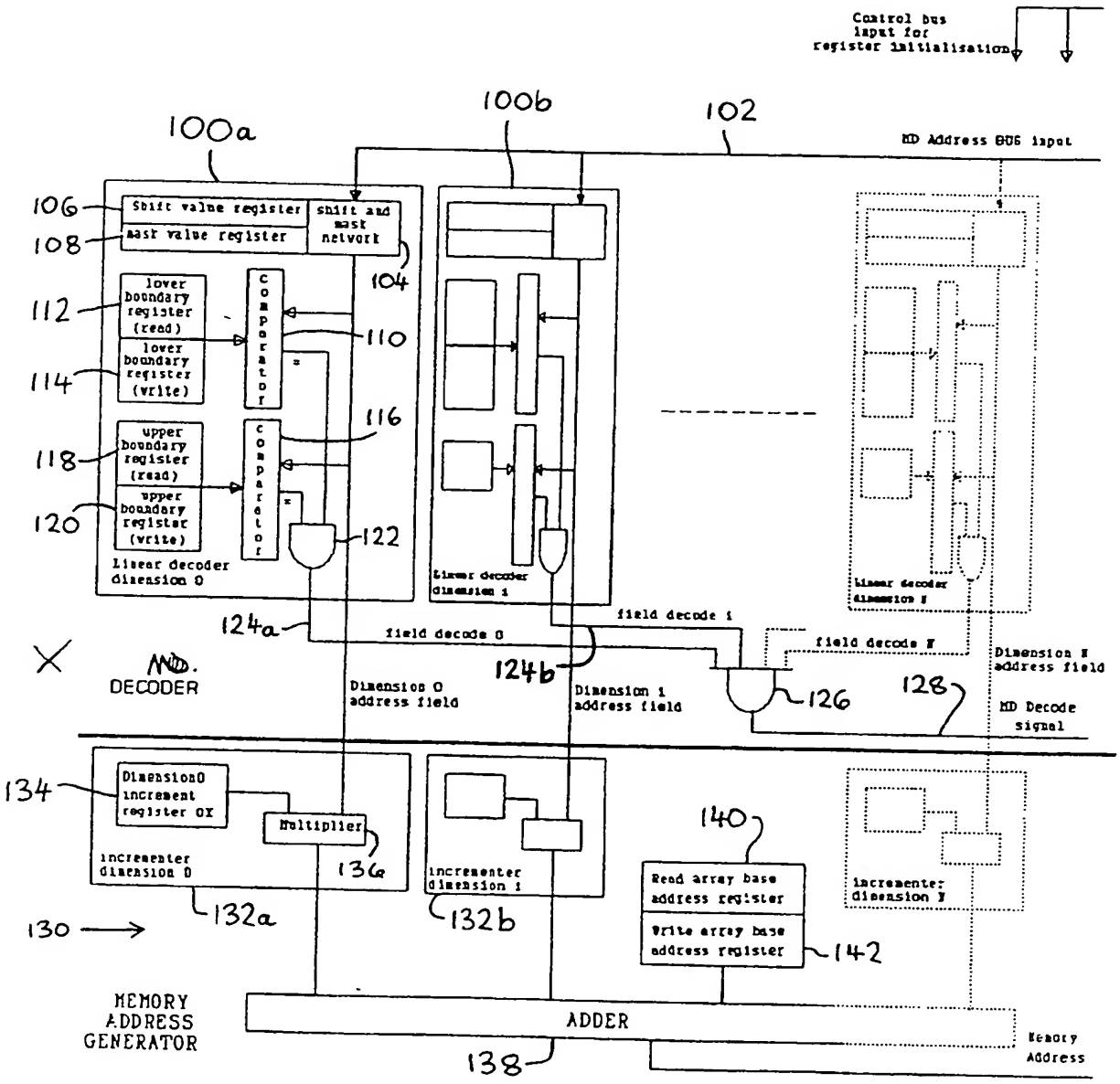


FIG. 8

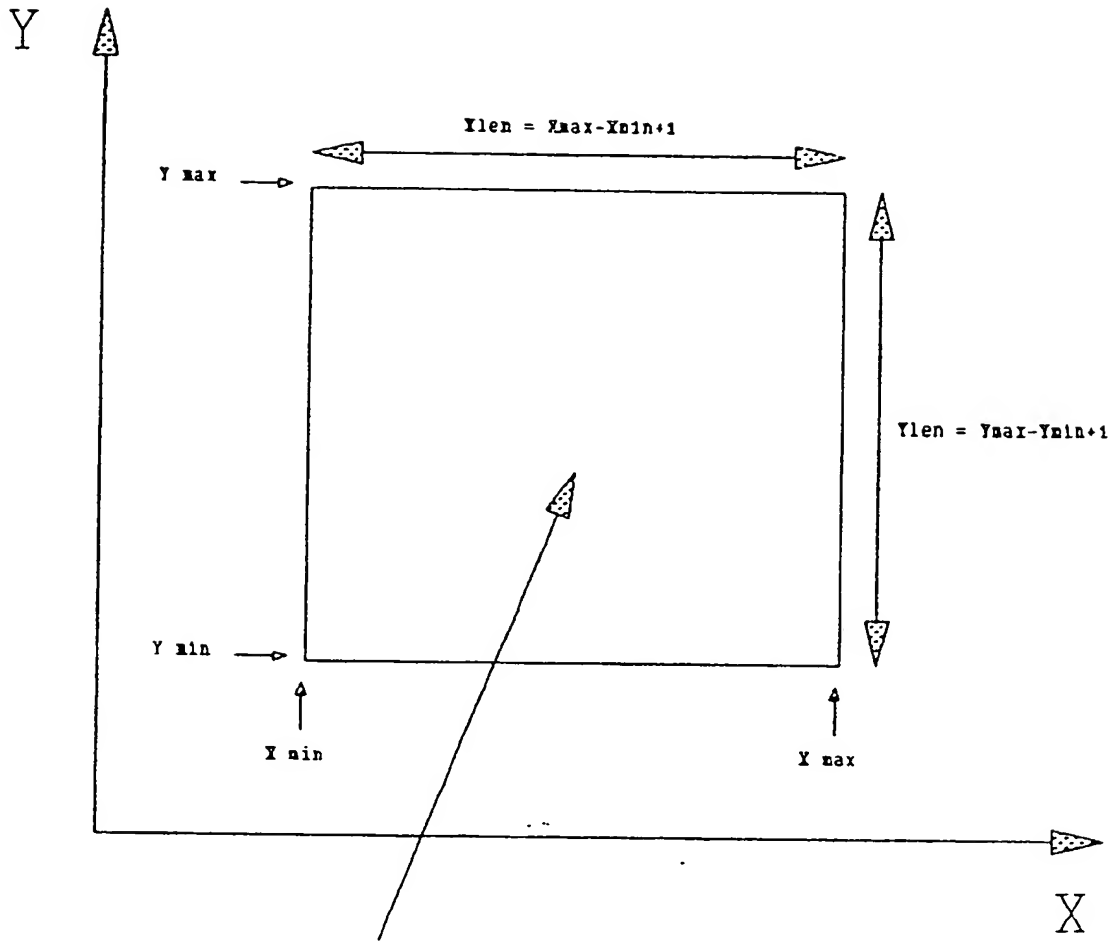

Y
Xlen = Xmax-Xmin+1
Y max
Ylen = Ymax-Ymin+1
Y min
X min
X max
X
2 Dimensional addresses between the bounds
of Xmin and Xmax: and Ymin and Ymax
lie within this decode area

FIG. 9

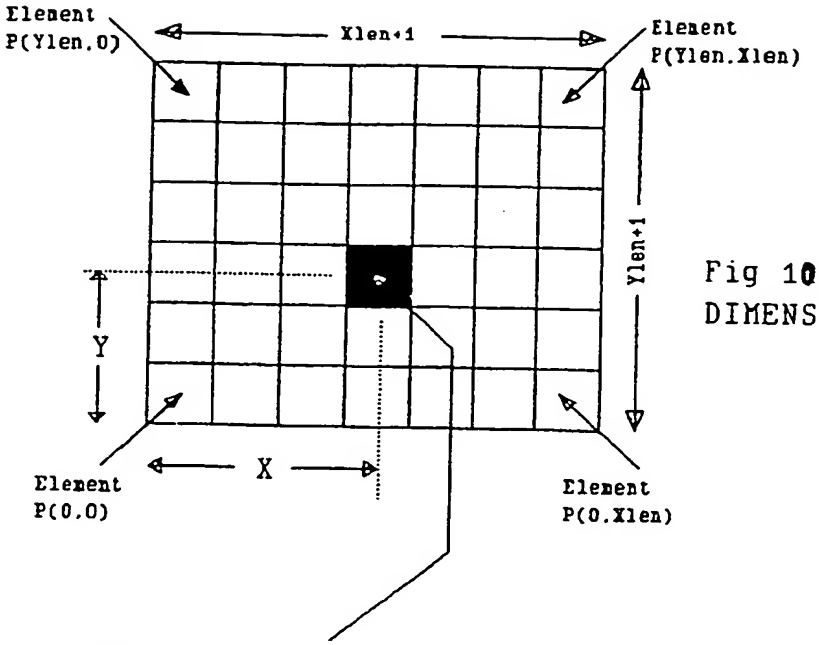


Fig 10a ARRAY P
DIMENSION P(Ylen,Xlen)

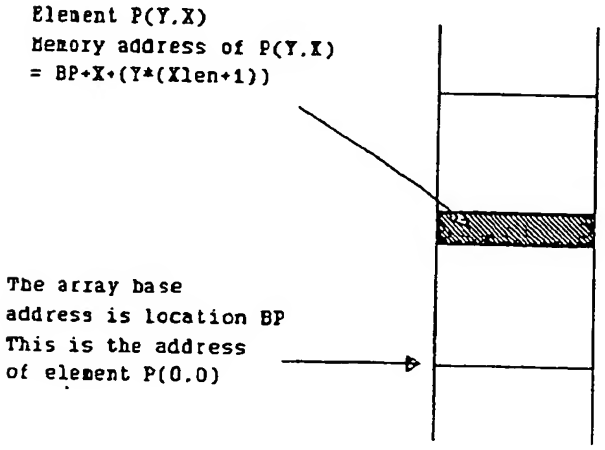


Fig 10b memory map
of ARRAY P when
held in linearly
addressed memory

FIG. 10

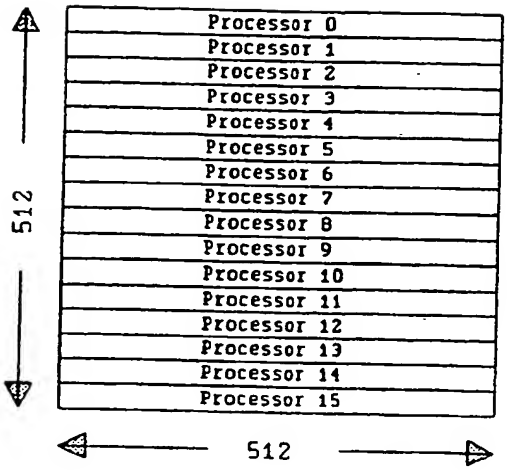


Figure 11a Buffer One format
Each processor stores
an array of size 2 x 32 x 512

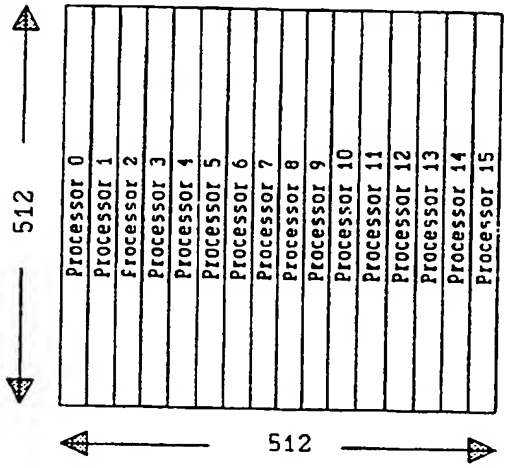


Figure 11b. Buffer Two Format
Each processor stores
an array of size 2 x 512 x 32

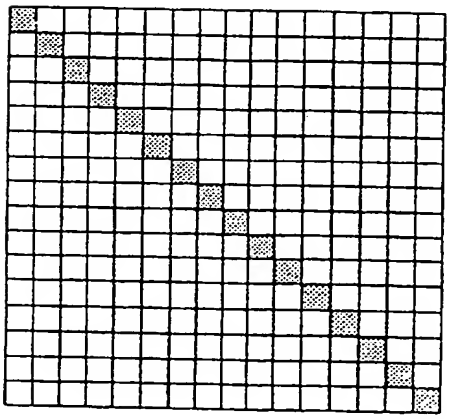


Figure 11c During Broadcast Bus transfer 256 data fragments of size 2 x 32 x 32 are automatically routed between the sixteen processors. The 16 fragments marked ▨ have the same source and destination processor.

Data storage formats for 2D FFT

FIG. 11

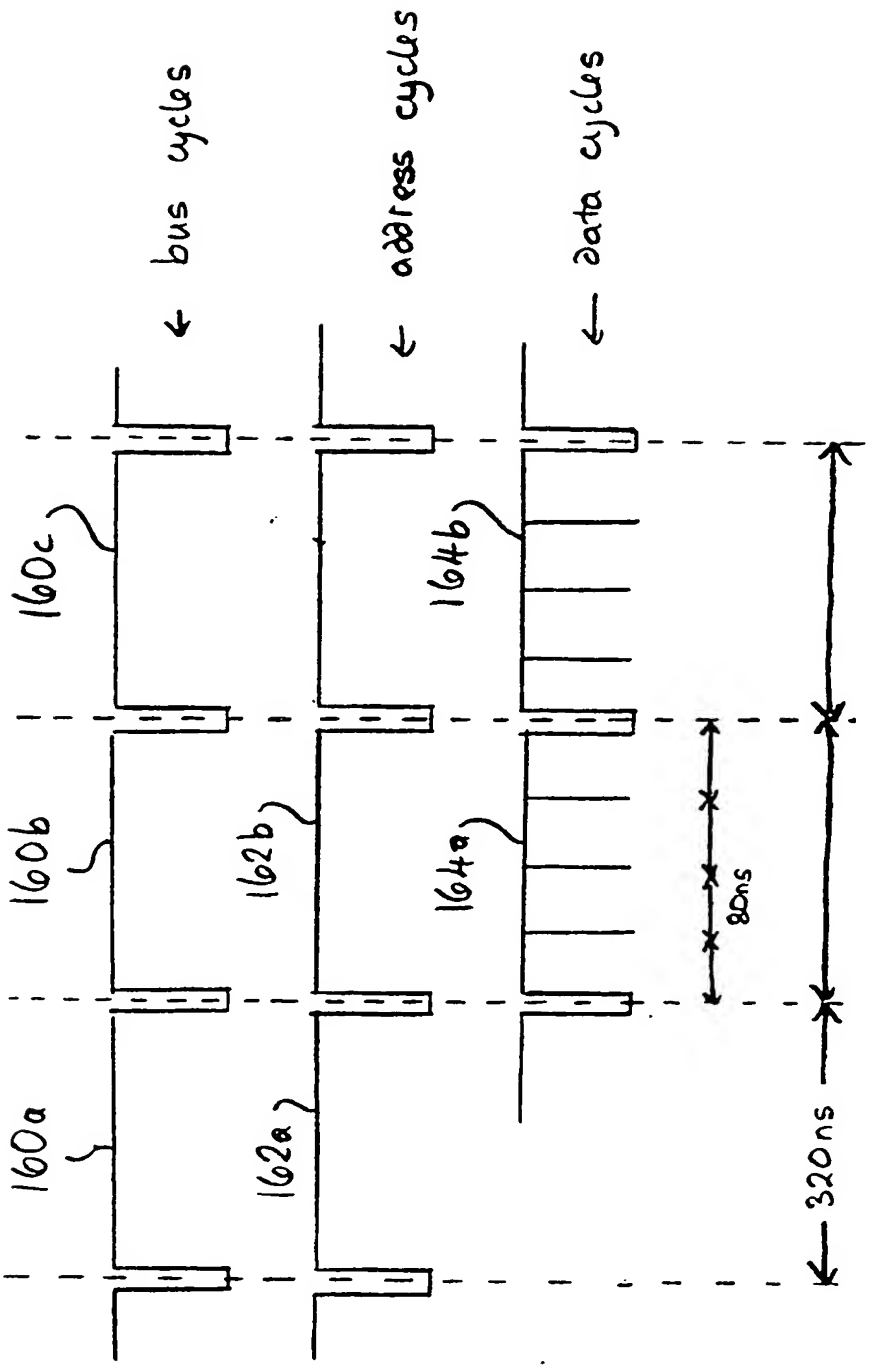


Fig. 12

European Patent Office

EUROPEAN SEARCH REPORT

Application Number

EP 89 30 7362

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
X	PROCEEDINGS OF THE FALL JOINT COMPUTER CONFERENCE 2-6 November 1986, page 1056-1063, Dallas, Texas, US; K. MORI et al.: "Autonomous Decentralized Soft ware Structure and its Application" * abstract; page 1057, chapter 3 - page 1059, chapter 4 *	1,13	G 06 F 15/80 G 06 F 13/38
A	idem	2-12,14	
X	N.E.C. RESEARCH & DEVELOPMENT no. 78, July 1985, pages 83-90, Tokyo, JP; N. ITO et al.: "NEDIPS: A Non-von Neumann High-Speed Computer" * page 83, chapter 2 - page 85, chapter 4 *	1,13	
A	idem	2-12,14	
X	GB-A-2 168 182 (CONIC CORP.) * abstract; page 1, line 74 - page 2, line 24; claims *	1,13	TECHNICAL FIELDS SEARCHED (Int. Cl.5)
A		2-12,14	G 06 F 15/80 G 06 F 13/38 G 06 F 9/44 G 06 F 9/46
A	MICROPROCESSORS & MICROSYSTEMS vol. 11, no. 5, June 1987, pages 255-263, London, GB; A. AYYAD et al.: "Multiprocessor scheme with application to macro-dataflow" * whole document *	1-14	
A	COMPUTER DESIGN vol. 27, no. 18, 1 October 1988, pages 20-26, Littleton, MA, US; T. WILLIAMS: "Software machine model blazes trail for parallel processing" * whole document *	1,13	

-/-

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
BERLIN	16-03-1990	DURAND J.

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P0401)

European Patent Office

EUROPEAN SEARCH REPORT

Application Number

EP 89 30 7362

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl. 5)
A	IEEE THE 15TH CONFERENCE OF ELECTRICAL & ELECTRONICS ENGINEERS IN ISRAEL 7-9 April 1987, pages 1.1.6(2)-1.1.6(4), Tel Aviv, Israel; R. GINOSAR et al.: "Topological Comparison of Static Multiprocessing Networks" * whole document * ---	1,13	
A	US-A-4 051 551 (D.H. LAWRIE et al.) * whole document * -----	1,13	
			TECHNICAL FIELDS SEARCHED (Int. Cl.5)

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
BERLIN	16-03-1990	DURAND J.

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P0401)